**Multiple and Multilevel Inheritance**

**LAB # 06**

** Fall 2019**

**Fall 2021**

**CSE208L Object oriented programming**

Submitted by: **Ashfaq Ahmad**

Registration No. : **19PWCSE1795**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Sumayyea salahuddin**

January 14, 2021

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objectives of the Lab:**

Objectives of the lab are to:

# Understand the concept of multiple and multilevel inheritance.

# Write two level inherited classes.

* Write a class inherited from multiple base classes.
* Write multi-file programs covering inheritance.

# Activity # 01

**Create a class First. It contains one protected data member f and one public input function f\_input(). Use the function to take f from user on runtime. Next, create a derived class Second from First class. This class also contains only one protected data member s and one public input function s\_input(). Call f\_input() function inside s\_input() and then take s from user on runtime. Finally, create another derived class Third from Second class. This class contains one protected data member t. It contains three public functions. An input function t\_input() that takes t from user on runtime, a max function max() that finds maximum of f, s, and t and displays the maximum, and show function that displays f, s, and t. Note, call s\_input() inside t\_input() and then take t from user. Write main function to test the functionality. Create an object of Third. Call t\_input(), show(), and max() functions according to test case given in 6.4.**

**Note: For python, keep same name for input function i.e. in1() in all three classes.**

**IN C++**

**SOURCE CODE:**

#include <iostream>

using namespace std;

class First

{

protected:

int f;

public:

void f\_input()

{

cout<<"Enter value of f: ";

cin>>f;

}

};

class Second: public First

{

protected:

int s;

public:

void s\_input()

{

f\_input();

cout<<"Enter value of s: ";

cin>>s;

}

};

class Third: public Second

{

protected:

int t;

public:

void t\_input()

{

s\_input();

cout<<"Enter value of t: ";

cin>>t;

}

void Max()

{

if(f>s && f>t)

cout<<"Max: "<<f<<endl;

else if (s>f && s>t)

cout<<"Max: "<<s<<endl;

else if (t>f && t>s)

cout<<"Max: "<<t<<endl;

}

void Show()

{

cout<<"f: "<<f<<endl;

cout<<"s: "<<s<<endl;

cout<<"t: "<<t<<endl;

}

};

int main()

{

Third t1;

t1.t\_input();

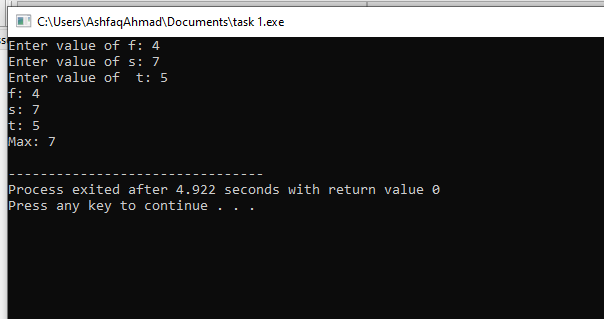
t1.Show();

t1.Max();

return 0;

}

**OBSERVATION**



**In Python**

**Source code:**

class First:

    f = 0

    @classmethod

    def input\_f(cls):

        First.f = float(input("Enter f :"))

class Second (First):

    s = 0

    @classmethod

    def input\_s(cls):

        First.input\_f()

        Second.s = float(input("Enter s : "))

class Third (Second):

    t = 0

    @classmethod

    def input\_t(cls):

        Second.input\_s()

        Third.t = float(input("Enter t : "))

    def show(self):

        print(f" Value of F : {First.f}\n"

        f"Value of S : {Second.f}\n"

        f"Value of t : {Third.t}")

    def max(self):

        max\_no = max(Third.t, Second.s, First.f)

        print(f"Max : {max\_no}")

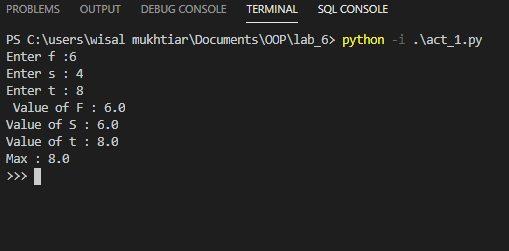
t = Third()

t.input\_t()

t.show()

t.max()

**OBSERVATION (TESTING & DEBUGING):**



# Activity # 02

Create a class base. It contains one protected data member ba and two public functions input\_base()and show\_base(). Use input\_base() to take ba from user on runtime while show\_base() to display content of ba. Create another class exponent. It also contains one protected data member exp and two public functions input\_exp() and show\_exp(). Use input\_exp() to take exp from user on runtime while show\_exp() to display content of exp. Next, create derived class power from base class and exponent class. This class contains one data member po. It contains three public functions. A constructor to initialize po with 1, an input function in1(), and show1() function. The in1() calls input\_base() and input\_exp() functions. The show1() calls show\_base() and show\_exp() functions; computes power using ba and exp and store in po; and displays computed power. Write main function to test the functionality. Create an object of power. Call in1() and show1()functions according to test case given in 6.4. Note: Write code for C++ and Python for this activity. Java does not support multiple inheritance.

**IN C++**

**SOURCE CODE:**

#include <iostream>

using namespace std;

class Base

{

protected:

double ba;

void input\_base()

{

cout<<"Enter base : ";

cin>>ba;

}

double show\_base()

{

return ba;

}

};

class Exponent

{

protected:

double exp;

public:

void input\_exp()

{

cout<<"Enter Exponent: ";

cin>>exp;

}

double show\_exp()

{

return exp;

}

};

class Power: public Base, Exponent

{

protected:

int po;

public:

Power():po(1){}

public:

void in1()

{

input\_base();

input\_exp();

}

void show1()

{

cout<<"Base: "<<show\_base()<<endl;

cout<<"Exponent: "<<show\_exp()<<endl;

for(int i{0}; i<show\_exp();i++)

{

po=po\*show\_base();

}

cout<<"Power = "<<po<<endl;

}

};

int main()

{

Power p1;

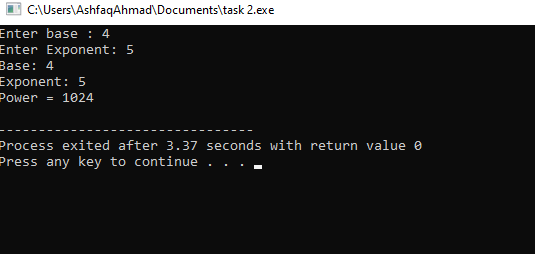
p1.in1();

p1.show1();

return 0;

}

**OBSERVATION (TESTING & DEBUGING):**



**In Python**

**Source code:**

from math import pow

class Base:

    ba = 0

    @classmethod

    def input\_base(cls):

        Base.ba = float(input("Enter base : "))

    @classmethod

    def show\_base(cls):

        return Base.ba

class Exponent:

    ex = 0

    @classmethod

    def input\_expo(cls):

        Exponent.ex = float(input("Enter expo : "))

    @classmethod

    def show\_expo(self):

        return Exponent.ex

class Power(Base, Exponent):

    Po = 0

    @classmethod

    def in1(cls):

        ba = Base.input\_base()

        ex = Exponent.input\_expo()

    @classmethod

    def show(cls):

        print(Base.show\_base())

        print(Exponent.show\_expo())

    @classmethod

    def calc\_po(cls):

        Power.po = pow(Base.ba, Exponent.ex)

        print(f"Power is : {Power.po}")

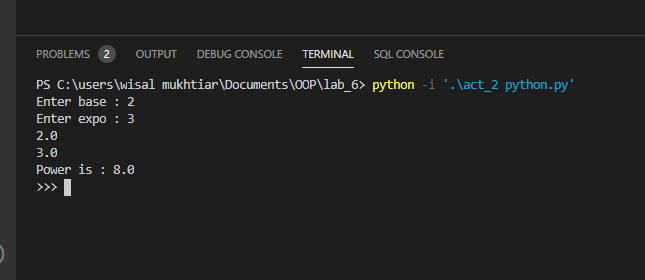
p1 = Power()

p1.in1()

p1.show()

p1.calc\_po()

**OBSERVATION (TESTING & DEBUGING):**



# Activity # 03

Redo Activity 6.3.1 and 6.3.2 using multi-file programming.

Note: In C++, create header file (\*.h) for each class and main file for main function (e.g. lab6t3.cpp). Include the header file to access the respective class.

Note: In python, save all class in separate \*.py file (e.g. lab6t1.py containing First, Second, and Third

classes and lab6t2.py containing base, exponent, and power classes )and then access using import in main python file (e.g. lab6t3.py).

Note: Since, Java is already doing multi-file programming so no need to do this activity in Java.

**Activity 2(Multi-File)(C++):**

**HEADER FILE (**classes.h)

#ifndef BASE\_H\_INCLUDED

#define BASE\_H\_INCLUDED

#include <iostream>

using namespace std;

class Base

{

protected:

double ba;

Public:

void input\_base();

void show\_base();

};

class Exponent

{

protected:

double exp;

public:

void input\_exp();

void show\_exp();

};

class Power: public Base, Exponent

{

protected:

int po;

public:

Power();

void in1();

void show1();

};

#endif

**Main File (main.cpp)**

#include <iostream>

#include <Classes.h>

using namespace std;

void Base::input\_base()

{

cout<<"Enter base : "<<endl;

cin>>ba;

}

double Base :: show\_base()

{

return ba;

}

void Exponent :: input\_exp()

{

cout<<"Enter Exponent: ";

cin>>exp;

}

double Exponent :: show\_exp()

{

return exp;

}

Power :: Power():po(1){}

void Power :: in1()

{

input\_base();

input\_exp();

}

void Power :: show1()

{

cout<<"Base: "<<show\_base()<<endl;

cout<<"Exponent: "<<show\_exp()<<endl;

for(int i{0}; i<show\_exp();i++)

{

po=po\*show\_base();

}

cout<<"Power = "<<po<<endl;

}

int main()

{

Power p1;

p1.in1();

p1.show1();

return 0;

}

**Activity 2 (Multi-file):**

**In Python**

**Source code:**

Main:

from classes import Third

t = Third()

t.input\_t()

t.show()

t.max()

Classes:

class First:

    f = 0

    @classmethod

    def input\_f(cls):

        First.f = float(input("Enter f :"))

class Second (First):

    s = 0

    @classmethod

    def input\_s(cls):

        First.input\_f()

        Second.s = float(input("Enter s : "))

class Third (Second):

    t = 0

    @classmethod

    def input\_t(cls):

        Second.input\_s()

        Third.t = float(input("Enter t : "))

    def show(self):

        print(f" Value of F : {First.f}\n"

        f"Value of S : {Second.f}\n"

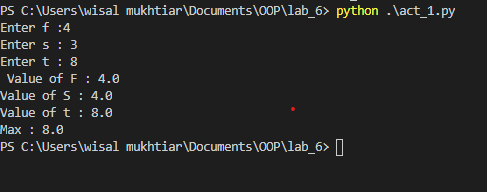
        f"Value of t : {Third.t}")

    def max(self):

        max\_no = max(Third.t, Second.s, First.f)

        print(f"Max : {max\_no}")

**OBSERVATION (TESTING & DEBUGING):**

****

**THE END**